design ideas

Edited by Bill Travis and Anne Watson Swager

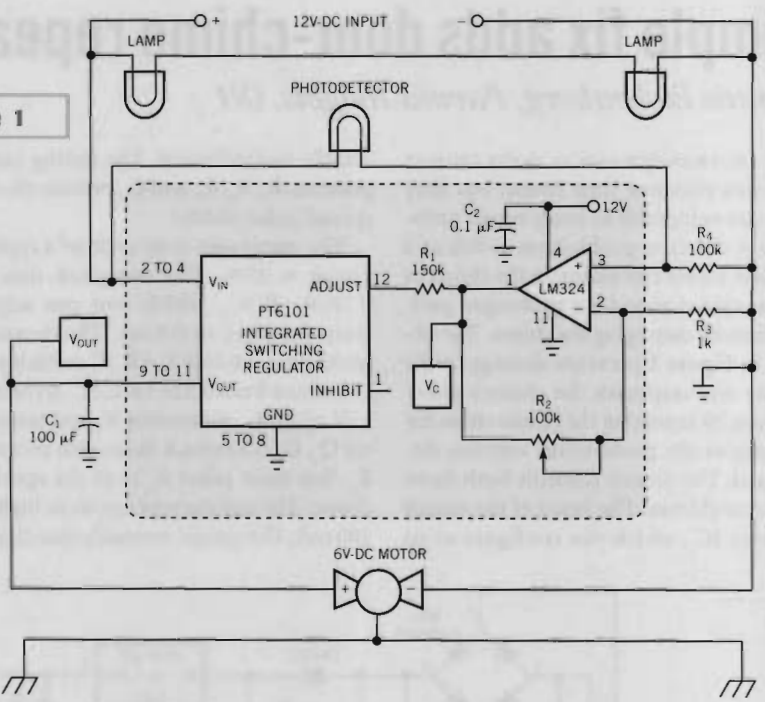
Switching regulator drives robot motor

Donald Comiskey, Power Trends Inc, Warrenville, IL

THE CIRCUIT IN **Figure 1** shows an example of how you can use an integrated switching regulator (ISR) to efficiently vary the speed of a permanent-magnet dc motor. In this application, the rotating head of a robot senses the presence of an oncoming object (perhaps a human). If the robot senses an object, it slows the rotation of its head to closely survey the surroundings. If an object is indeed present and comes within a certain distance of the robot, the robot's head stops rotating and "looks" in the direction of the object. The robot uses a typical proximity-detection device that senses reflected light (**Figure 2**). It's nose contains a photodetection device that senses the light that reflects from an oncoming object. The primary source of the light is the robot's eyes, which contain two lamps. The photodetector in the nose resides in a black tube to reduce the effects of ambient light.

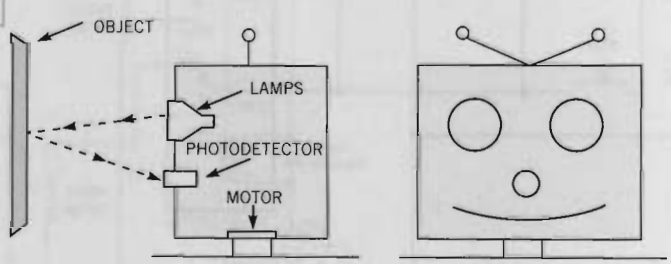
The sensed light causes a photodetection current to flow through the photodetector and R_4, developing a voltage across R_4 (**Figure 1**). The LM324 op amp, which you configure as a noninverting amplifier with an adjustable gain of

Figure 1



This robotic circuit uses an integrated switching regulator as a power op amp to control the rotation of an object-seeking robot's head.

Figure 2



R2D2 has lamps in his eyes to provide a light source and a photodetector in his nose to detect light reflected from an oncoming object.

$1+R_2/R_3$, amplifies this voltage. The amplified voltage is a control voltage, V_C, which routes to the ISR's Adjust pin via R_1. A decrease in V_C, relating to a decrease in light level, causes the ISR's output voltage and the corresponding speed of the dc motor to increase. An increase in V_C, relating to an increase in light level, causes V_{OUT} and the corresponding motor speed to decrease. A further increase in light level causes V_C to increase to a point at which V_{OUT} becomes low enough to stop the rotation of the motor. A linear relationship exists between V_C and V_{OUT}: $V_{OUT}=-V_C+6.25V$.

In the absence of light, the output voltage of the op amp saturates near 0V. The equation shows that a control voltage of 0V produces an ISR output voltage of 6.25V, causing the motor to rotate at its maximum speed. A light level that produces a 6.25V control voltage results in an ISR output voltage of 0V, causing the motor to stop rotating. Intermediate motor speeds result from control voltages between the extremes of 0 and 6.25V. You can adjust feedback resistor R_2, which sets the op-amp gain, to obtain any desired sensitivity. The PT6101 ISR in **Figure 1** is 85 to 90% efficient and supplies motor currents as high as 1A. (DI #2353).

To Vote For This Design,
Circle No. 311

Simple fix adds door-chime repeater

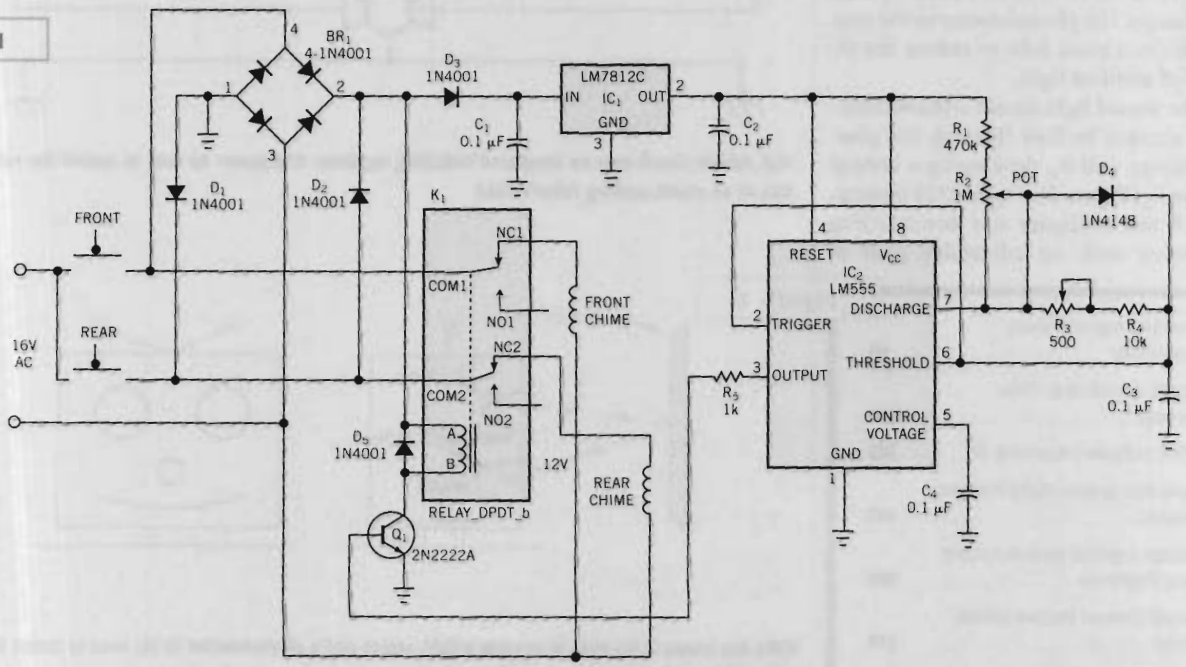
Dennis Eichenberg, Parma Heights, OH

Electromechanical door chimes can enhance your home, but they are vulnerable to costly repair problems. A defective pushbutton switch or a careless visitor can maintain the chime in an energized state for a prolonged period, thereby damaging the chime. The circuit in **Figure 1** prevents damage to the chime and improves the chime's effectiveness by repeating the chime strike for as long as the pushbutton remains depressed. The circuit controls both front and rear chimes. The heart of the circuit is timer IC_2, which you configure as an astable multivibrator. The timing components, R_1, R_2, R_3, and C_3, provide the required pulse widths.

The maximum duty cycle of a typical chime is 25%. The energized time is $0.76(R_3+R_4)C_3$, which you can adjust from 7.6 msec to 0.4 sec. The de-energized time is $0.693(R_1+R_2)C_3$, which you can adjust from 0.3 to 1 sec. IC_2 drives the 12V relay, K_1, via resistor R_5 and transistor Q_1. D_5 is a flyback diode that protects K_1. You must select K_1 to fit the specific chime. The coil current can be as high as 200 mA. The circuit normally uses closed contacts so that illuminated pushbutton switches operate properly. The 16V-ac door-chime power energizes the circuit via bridge BR_1. IC_1 provides voltage regulation, and C_1 and C_2 provide filtering. Diodes D_1 and D_2 provide power from the rear-door pushbutton. D_3 isolates the filtered timer power from the relay. (DI #2349).

To Vote For This Design,
Circle No. 312

Figure 1



You can both protect your door chime and obtain pleasing chime repetition with this circuit.

Clock multiplier circumvents PLL

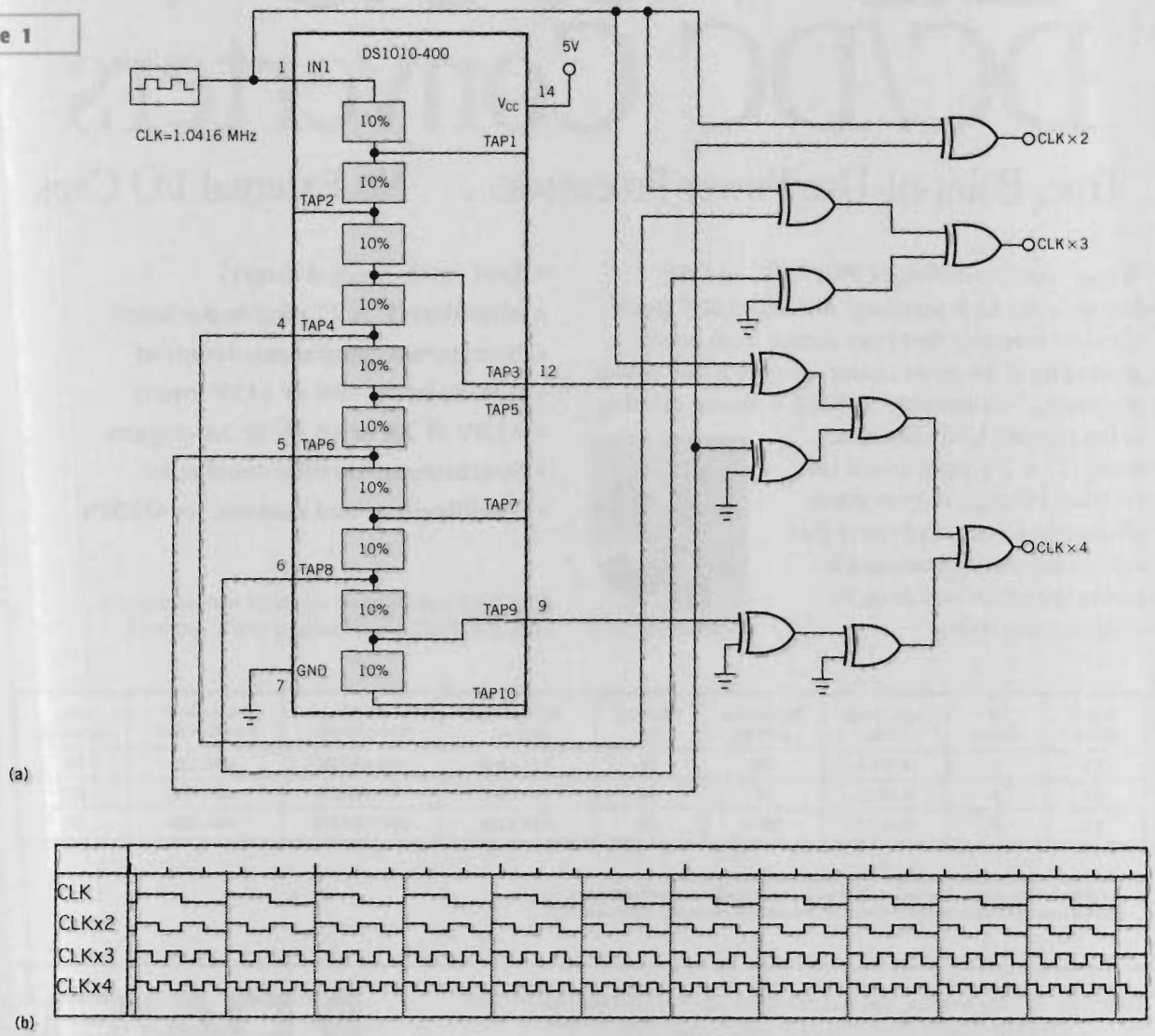
Jose Carlos Cossio, Santander, Spain

USING A STANDARD PLL CIRCUIT, such as the CMOS 4046B with some passive components, is a well-known way to design a clock multiplier. Unfortunately, using a PLL in a digital circuit has two disadvantages: It takes a long time for the circuit to reach a stable output frequency, and the frequency-drift compensation has a complex design.

An alternative clock-multiplier approach uses simple combinatorial logic and a delay line (**Figure 1**). By taking into account the propagation delays in combinatorial logic and the fact that most logic designs are edge-triggered, the circuit can multiply by 2, 3, 4, and potentially beyond. You can use this technique to increase the system clock speed and maintain a low-frequency clock for the rest of the circuit, thereby helping to reduce EMI and cost in high-speed logic designs. Also, the circuit exhibits a very fast start-up. This approach is common in modern µP architectures that use clock-doubling techniques to achieve better performance at low costs.

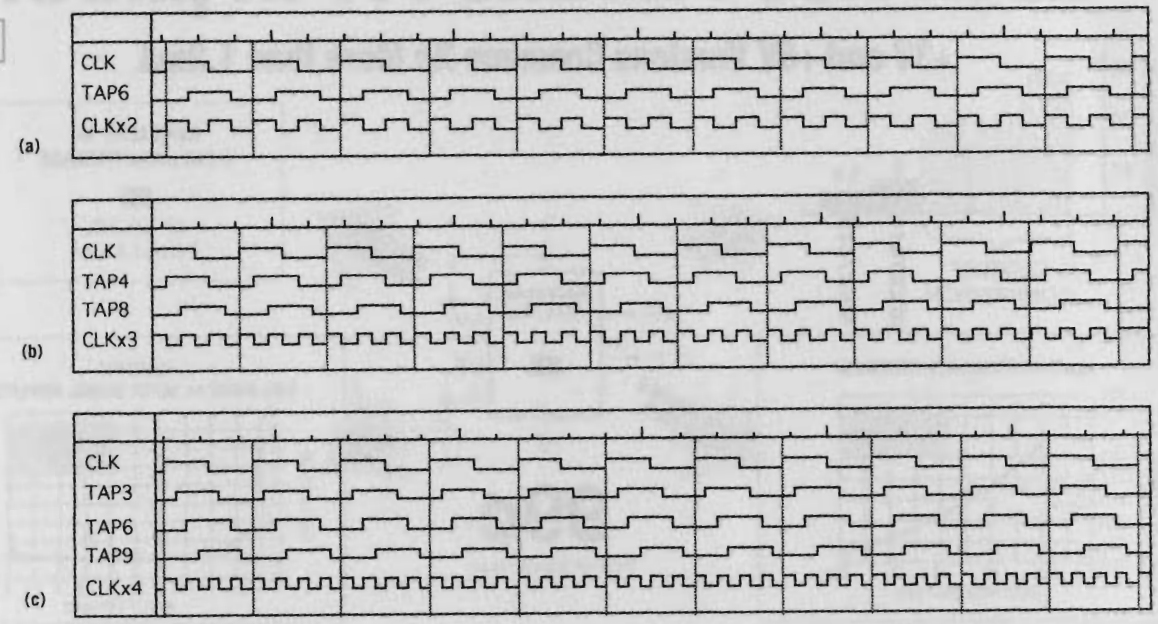
In place of the silicon delay line, you could use standard CMOS logic. However, the differing propagation delays for high-to-low and low-to-high transitions

Figure 1



Using a delay line and a phase-comparison scheme with XOR gates (a), this circuit multiplies the input clock by 2, 3, and 4 without a PLL (b).

Figure 2



Internally delayed clock signals produce the desired clock-multiplication factors of 2× (a), 3× (b), and 4× (c).

($t_{pdHL} \neq t_{pdLH}$) presents a problem. A standard delay line with fixed and known delay taps is preferable.

The circuit in **Figure 1a** uses a Dallas Semiconductor (www.dalsemi.com) DS-1010-400 delay line, which has 10 built-in fixed delays of 40 nsec, to implement a 2×, 3×, and 4× clock multiplier that operates from an external 1.0416-MHz clock. This design requires XOR gates with short propagation delays, such as advanced-bipolar or fast CMOS gates with propagation delays of less than 10 nsec, so that the XOR-gate delays are negligible.

The circuit works by using the XOR gates to perform a phase comparison. These gates monitor the difference between the incoming clock signal and the delayed clock signals. First, the circuit shifts right, or delays, the signal to be multiplied: once for 2×, twice for 3×, three times for 4×, and so on. Then, because an XOR gate sits between the source and the delayed signals, the result of the XOR operation is the signal multiplied by 2, 3, and 4. The resultant clock signals have the same duty cycle as the incoming signal (**Figure 1b**). **Figure 2** shows the internally delayed clock signals that are necessary to produce the 2×, 3×, and 4× clocks.(DI #2344).

Synthesize optimal digital-frequency dividers

Lindo St Angel, PrairieComm Inc, Arlington Heights, IL

For many applications, you need to divide a reference clock into one or more subclocks to use in different parts of the system. Sometimes, this divider circuit is simple. For example, a circuit that divides by an integer number is easy to construct using a few flip-flops and assorted logic gates. However, often you must build a noninteger divider, for example, a divide by 7/8 or 512/1025. In these cases, you could use a divider and a PLL to divide the input reference by the denominator and then multiply the result by the numerator, but the circuit would become relatively complex and require analog components.

However, if your application can tolerate some clock jitter, there is another answer: the binary-rate-multiplier (BRM) circuit. This well-known circuit works by multiplexing two dividers into and out of the divide path. Unfortunately, it is sometimes difficult to find the set of design parameters that gives the least amount of jitter and the desired divider ratio. Using the following method and circuit you can easily generate optimum BRMs.

The average frequency, f_{OUT}, of the BRM output is:

$$f_{OUT} = f_{IN} \times \frac{(REP_1 + REP_2)}{(REP_1 \times DIV_1) + (REP_2 \times DIV_2)}, \quad (1)$$

where f_{IN} is the frequency of the input

reference clock, DIV_1 is the number of times the first divider divides the input clock, DIV_2 is the number of times the second divider divides the input clock, REP_1 is the number of input clock cycles for which the first divider is active, and REP_2 is the number of input clock cycles for which the second divider is active.

Instantaneous frequency of the BRM output equals the frequency of the input clock divided by the the active divider. The BRM output jitters between these two frequencies. However, you can minimize this jitter by choosing the parameters DIV_1 and DIV_2 according to

$$DIV_1 = INT\frac{f_{IN}}{f_{OUT}}, \quad (2)$$

and

$$DIV_2 = DIV_1 + 1. \quad (3)$$

You then need to choose values for the parameters REP_1 and REP_2. One way to choose these values is to transform **Equation 1** into an optimization problem and finding the values of REP_1 and REP_2 that minimize the transformed equation, given your chosen values for DIV_1 and DIV_2:

$$\min Z = \frac{f_{IN}}{f_{OUT}} \times \frac{(REP_1 + REP_2)}{(REP_1 \times DIV_1) + (REP_2 \times DIV_2)} - 1. \quad (4)$$

The constraints on this equation are that Z is greater than or equal to zero and that REP_1 and REP_2 are integers that are greater than or equal to unity.

You can solve **Equation 4** using integer nonlinear optimization techniques. Many commercial software packages, including Microsoft Excel, solve these classes of problems. Enter **Equation 4** into Excel and use the solver from the tools menu to operate on the equation, using the above constraints.

To get Z to exactly equal zero, it is important to increase the solver's precision and tolerance settings to about 10 times their default values. Also, make sure that you check your results by plugging in all values and checking that Z exactly equals zero. If it doesn't, you have to increase the solver's precision and tolerance settings.

Once you have obtained values for all the parameters, you can use them in the Verilog model to get a circuit from a logic-synthesis tool (**Listing 1**). If you don't have access to logic synthesis, the Verilog model can give you some idea of how to manually build up the circuit from gates. You can download **Listing 1** from *EDN*'s Web site, www.ednmag.com. Click on "Search Databases/Links Page," and then enter the Software Center to download the file for Design Idea #2358. (DI #2358).

LISTING 1—BINARY-RATE-MULTIPLIER VERILOG

```
// Ports:
//
// clkin:     Clock input to be divided down.
// rstb:      Reset, active Low input.
// clkout:    Binary Rate Multiplied clock signal output.
//
module brm (clkin, rstb, clkout);

//
// Change parameters below to yield the desired BRM.
//
// You can do this from the module which instanciates the BRM by:
//       inst_brm #(div1, rep1, div2, rep2) brm;
//
// The divide ratio is equal to:
// (rep1 + rep2)/(rep1*div1 + rep2*div2)
//
// The default parameters give a 7/18 divide ratio.
//
   parameter div1 = 2; // Divider one
   parameter rep1 = 3; // Number of times to repeat divider 1
   parameter div2 = 3; // Divider two
   parameter rep2 = 4; // Number of times to repeat divider 2

//
// These parameters must be chosen so that synthesis will create a
// large enough counter. The mbit parameter is chosen so that two raised
// to the power of mbit is equal to or greater than the max of (div1, div2).
// The sbit parameter is chosen so that two raised to the power
// of sbit is equal to or greater than the max of (div1*rep1, div2*rep2).
//
   parameter mbit = 2; // Number of bits needed for main counter
   parameter sbit = 4; // Number of bits needed for state counter

   input clkin, rstb;

   output clkout;

   reg [(mbit - 1):0] mcnt;
   reg [(sbit - 1):0] scnt;
   reg clkout, state, rst_mcnt, rst_scnt;

   wire rst;

   assign rst = ~rstb;

//
// Main Counter.
//
   always @(posedge clkin or posedge rst)
      if (rst == 1'b1)
         mcnt = 0;
      else
         if (rst_mcnt == 1'b1)
            mcnt = 0;
         else
            mcnt = mcnt + 1;

//
// Main Counter reset generation.
// This is used to switch the modulus of the Main Counter.
//
   always @(mcnt)
      case (state)
         1'b0 : rst_mcnt = (mcnt == (div1 - 1)) ? 1'b1 : 1'b0;
       1'b1 : rst_mcnt = (mcnt == (div2 - 1)) ? 1'b1 : 1'b0;
      endcase

//
// Output Register
//
   always @(posedge clkin or posedge rst)
      if (rst == 1'b1)
         clkout = 1'b0;
      else
         clkout = rst_mcnt;

//
// State Counter and State Flag generation.
// When State Flag is Low, the BRM is using the first divide ratio,
// when State Flag is High, the BRM is using the second divide ratio.
//
   always @(posedge clkin or posedge rst)
      if (rst == 1'b1)
         begin
            scnt = 0;
         state = 1'b0;
       end
      else
         if (rst_scnt == 1'b1)
          begin
             scnt = 0;
              state = ~state;
          end
         else
          scnt = scnt + 1;

//
// State Counter reset generation.
// This is used to switch the modulus of the State Counter.
//
   always @(scnt)
      case (state)
         1'b0 : rst_scnt = (scnt == ((div1 * rep1) - 1)) ? 1'b1 : 1'b0;
        1'b1 : rst_scnt = (scnt == ((div2 * rep2) - 1)) ? 1'b1 : 1'b0;
      endcase
endmodule
```

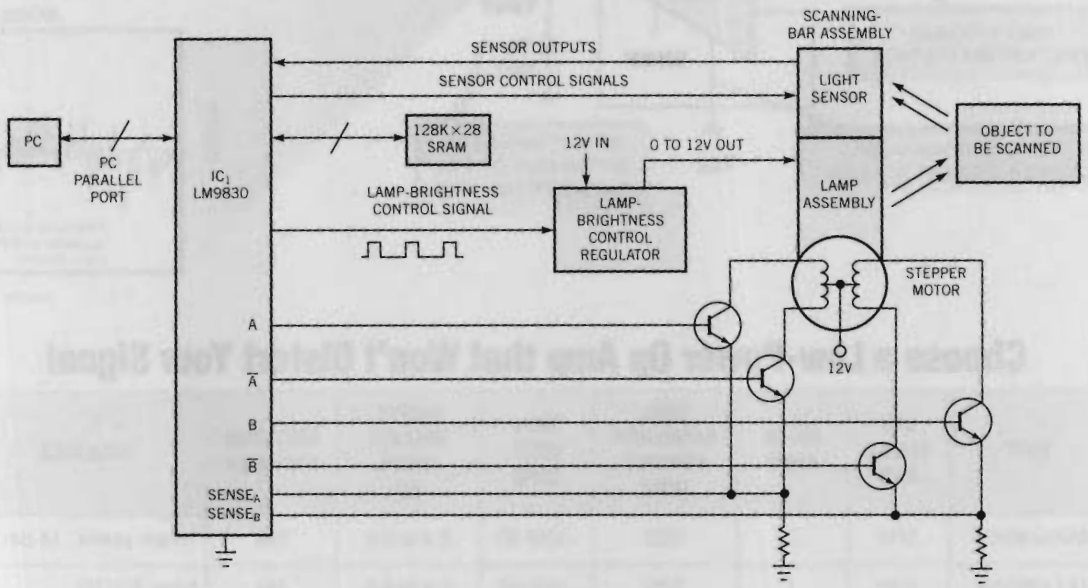
Voltage regulator controls scanner-lamp brightness

Chester Simpson, National Semiconductor, Santa Clara, CA

An image-scanning device performs best when the scanner IC controls the brightness of the image reflected into the light sensors. The best way to control brightness is by adjusting the intensity, or light output, of the fluorescent lamp that shines light onto the scanned item. Traditionally, adjustable brightness has been economically unfeasible in most scanners because it is costly to implement a high-current voltage source that is also software-programmable. However, you can use a simple, low-cost circuit to implement full-range lamp-brightness control (**Figure 1**).

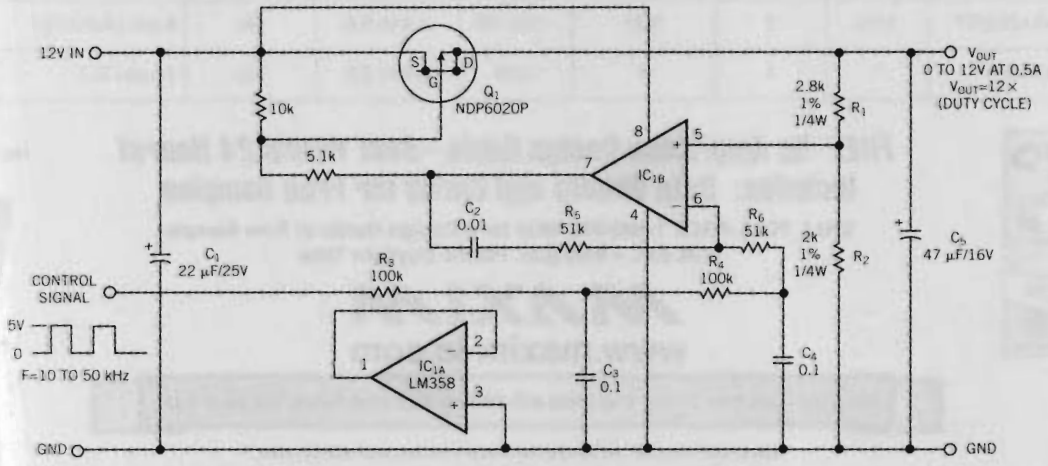
The fluorescent lamps in most low-cost scanners operate from a fixed 12V source, which operates the lamp at maximum brightness. If you reduce the source to a lower voltage, you can adjust the brightness of the lamp to virtually any level. You can accomplish the voltage change using a circuit that takes in 12V and produces a regulated dc output volt-

Figure 1



A lamp-brightness control regulator provides full-range brightness control depending on the duty cycle of the control signal from IC_1.

Figure 2



NOTES:
ALL RESISTORS ARE 5% TOLERANCE, 1/8W, UNLESS OTHERWISE INDICATED.
C_1 AND C_5 ARE SOLID TANTALUM. ALL OTHER CAPACITORS ARE CERAMIC.
SOLDER Q_1 TO PC BOARD WITH COPPER AREA ≥1 $IN.^2$ FOR HEAT SINKING.

This control regulator averages the control signal to provide an adjustable reference input to IC_{1B} that ultimately determines the level of V_{OUT}.

age that varies from 0 to 12V, depending on the lamp-brightness control signal coming from IC_1's scanner IC. IC_1 sets the duty cycle of the control signal, which is a 5V pulse train.

The lamp-brightness control regulator takes in 12V and regulates this input to any voltage from 0 to 12V (**Figure 2**). Because the on-resistance of the NDP6020P FET is only approximately 50 mV, the maximum regulated output voltage the circuit provides is within approximately 25 mV of the input voltage at the typical lamp current of 0.5A.

The circuit generates a regulated output using error-amplifier IC_{1B}, the R_1/R_2 resistive divider, and an adjustable reference voltage at Pin 6 of IC_{1B}. The circuit produces this reference voltage by averaging the input control signal's square-wave pulse train. R_3, C_3, R_4, and C_4 filter the square waves into a dc voltage. The voltage across C_4 is the average value of the pulse train, which is directly proportional to the duty cycle:

$$V_{C4} = 5V \times \text{DUTY CYCLE}.$$

By adjusting the duty cycle of the 5V pulse train, you can linearly vary the reference voltage at Pin 6 of IC_{1B} from 0 to 5V. You can program a maximum of 4095 duty-cycle values for the LM9830 scanner IC, which yields 4095 brightness levels.

IC_{1B} compares the voltage across C_4 with the voltage at the center of R_1 and R_2, which the circuit derives from V_{OUT}. The regulating action of IC_{1B} constantly adjusts the gate-drive voltage to Q_1, forcing V_{OUT} to a value that keeps equal the voltages at the inputs of IC_{1B}. In this way, the voltage at C_4, which is proportional to the duty cycle of the pulse train, controls the regulated voltage, V_{OUT}. You can calculate V_{OUT} using the following equation:

$$V_{OUT} = V_{C4} \times \frac{R_1 + R_2}{R_2} = V_{PK} \times \text{DUTY CYCLE} \times \frac{R_1 + R_2}{R_2},$$

where V_{PK} is the peak amplitude of the pulse train (5V in this application) and duty cycle is the pulse on-time divided by the total period.

R_5, R_6, C_1, C_2, and C_5 are necessary for compensation and stability. For the component values in the **figure**, the best performance occurs when the frequency of the pulse train is 10 to 50 kHz. (DI #2356).

To Vote For This Design,
Circle No. 315

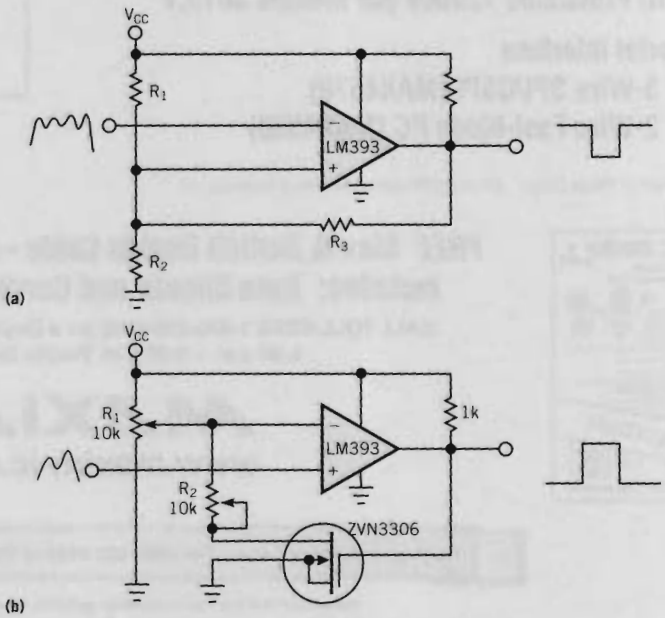
Simple changes improve Schmitt trigger

Ron Patrick, ECM, Los Altos, CA

The classical Schmitt trigger circuit that you'll find in textbooks provides some noise immunity, such as for the front end of magnetic pickup in position-sensor circuits (**Figure 1a**). You adjust the two threshold levels using R_1 and R_2 or R_2 and R_3. The circuit works acceptably but has two problems. First, each resistor affects both levels, making the circuit adjustment an iterative process. Second, the op amp's current-sinking ability limits your choices for R_3 and, hence, your ability to adjust the circuit. The improved trigger circuit in **Figure 1b** has independently adjustable levels; you first adjust R_1 and then R_2. The circuit also eliminates the sensitivity to the op amp's current-sinking capability. (DI #2361).

To Vote For This Design,
Circle No. 316

Figure 1



The classical Schmitt-trigger circuit (a) requires an iterative adjustment and the op amp's sink current affects the ability to adjust the circuit. The adjustable levels of an improved circuit (b) are completely independent.